BECCA: Reintegrating AI for Natural World Interaction

Brandon Rohrer
Intelligent Systems, Robotics, and Cybernetics Group
Sandia National Laboratories
Albuquerque, New Mexico

Abstract

Natural world interaction, the pursuit of arbitrary goals in unstructured physical environments, is an excellent motivating problem for the reintegration of artificial intelligence. It is the problem set that humans struggle to solve. At a minimum it entails perception, learning, planning, and control, and can also involve language and social behavior. Although it may be impossible for one agent to perform well in the entire problem space of natural world interaction, an agent's fitness is indicated by being able to perform a wide variety of tasks. In order to address the problem of natural world interaction, a brain-emulating cognition and control architecture (BECCA) was developed. It uses a combination of feature creation and model-based reinforcement learning to capture structure in the environment in order to maximize reward. BECCA avoids making common assumptions about its world, such as stationarity, determinism, and the Markov assumption. BECCA has been demonstrated performing a set of tasks which is non-trivially broad, including a vision-based robotics task. Current development activity is focused on applying BECCA to the problem of general Search and Retrieve, a representative natural world interaction task.

Introduction

Unstructured robotics tasks are well suited to be challenge problems for big AI. Experience suggests that narrow problems are best solved by carefully engineered point solutions. However, these solutions tend not to be intelligent in the sense that they can't solve problems for which they weren't explicitly designed. There is widespread interest in more intelligent methods, but investigators' aesthetic pleasure in generality rarely overcomes the practical gains afforded by specificity. Quantitative performance improvements on a problem typically drive publication decisions, and hence research efforts, more strongly than statements about an algorithm's breadth. One way to encourage the development of intelligent methods is to focus on solving problems that require them by definition. An unstructured task is one about which few details are known beforehand. In order to solve it, an agent must in theory be capable of solving *every* task with which it might be presented. Focusing on physically embodied agents in robotics tasks also drives the integration of perception, learning, planning, and control. An appropriate challenge task transforms the abstract engineering virtues of integration and generality into concrete development requirements.

Natural World Interaction (NWI) is the set of all tasks in which a goal is pursued in a physical environment. It encompasses all the goal-pursuit tasks humans perform. Goals may be abstract and a function of time. Achieving them may involve speech, social interaction, game playing, and reasoning. Measuring performance on the entire set of NWI task space would be infeasible, but the task space can be sampled to obtain an estimate. Agents' performance can be compared against each other on a randomly selected set of tasks. An agent with mediocre performance on many tasks would be superior to an agent that had optimal performance on one task but very poor performance on all others. Physical interaction tasks performed by machines implies robotic embodiments of the software agents. Thus, NWI is a set of unstructured robotics tasks, and serves as a bold challenge problem for the integration of AI.

A brain-emulating cognition and control architecture (BECCA) is being developed in an attempt to address NWI. BECCA is designed with few assumptions built in, in order to maximize its applicability to as many tasks as possible. It is modeled after hypothesized function of human and animal information processing, since biological solutions are currently the state of the art in NWI.

There is a small but vital body of research in robots performing unstructured goal pursuit. (Beeson, Modayil, and Kuipers 2010; Konidaris et al. 2011; Sutton et al. 2011, for example) (Note that "unstructured" does not imply that there is no structure inherent in the task, but rather than the agent–and the agent's creator–is not privy to that structure beforehand.) The limited scope of the tasks performed in these efforts highlight the challenges involved, but their successes justify the research direction and give hope for large steps forward in the near term.

NWI lends itself to formulation as a reinforcement learning (RL) problem in the broad sense (Sutton and Barto 1998), that is, any solution to NWI accepts observations and produces actions to maximize a scalar reward signal, but may not rely on common RL techniques. BECCA is an RL agent in this broad sense. It couples a feature creator and a model-based reinforcement learner to achieve general goal pursuit behaviors. The feature creator identifies patterns in the input, allowing for a concise representation of observations. The stream of features passed to the reinforcement learner are its experiences. It uses them to create a model, which it then uses to make predictions and plan. It also learns which features tend to be associated with reward and directs its planning accordingly.

Method

A block diagram shows how BECCA's functional components interact. (See Figure 1.) BECCA issues action commands to the world and receives from the world a reward signal and observations, in the form of sensory input and basic features. Specifically, at each discrete time step, BECCA interfaces with the world in the following way:

- It reads in a sensory observation, a vector $\mathbf{o} \in \mathbb{R}^k | 0 \leq o_i \leq 1$.
- It reads in a basic feature observation, a vector $\mathbf{b} \in \mathbb{R}^m | 0 \leq b_i \leq 1$.
- It receives a reward, a scalar $r \in \mathbb{R} | -1 \leq r \leq 1$.
- It outputs an action, a vector $\mathbf{a} \in \mathbb{R}^n | 0 \leq a_i \leq 1$.

The three main functional entities of the agent system are the *feature creator* and the *reinforcement learner*, elements of BECCA, and the *world*, representing everything else.

Feature creator

The role of the feature creator is to twofold: 1) create a feature space into which inputs can be mapped and 2) map the input into that feature space at each time step. As shown in Figure 1, sensory inputs are formed into groups based on how often they are co-active, and patterns within each group are identified as features and added to its feature set.

Feature creation, as BECCA does it, is finding linear combinations of inputs that reflect the underlying structure of the world. This is distinct from feature selection, which is identifying a subset of inputs as being most significant and disregarding the rest. BECCA's feature creation may be considered a dimensionality reduction method, but not strictly so, since the number of features created may outnumber the original inputs.

Grouping Grouping is based on a modified correlation of input elements. The pseudo-correlation takes into account the fact that the co-activity of two input elements is meaningful, but co-inactivity is not. It also introduces asymmetry. If input A is active every time B is active, then B may be considered highly related to A, but if B is only active ten percent of the time that A is active, then A would be only weakly related to B.

The correlation is incrementally estimated at each time step. Once the average correlation between two elements exceeds a threshold, the two elements form the nucleus of a group. Additional elements are added to the group one at a time while the average correlation between the closest candidate element and the rest of the group exceeds a threshold. In this way, groups of correlated elements are formed automatically. Elements may become members of more than one group, but the level of correlation required increases with each additional group it joins.

The vector of input elements, $\mathbf{o}$, constitutes a k-dimensional space, $\mathbb{R}^k$. A group with p member elements constitutes a p-dimensional subspace of the input space, $\mathbb{R}^p$. Due to the constraints on the input elements ($0 \leq o_i \leq 1$), all combinations of observed group inputs fall within one quadrant of the space.

Feature creation A feature is represented by a unit vector in the group subspace. At each time step, the distance (in the sense of vector angle) between the group input and the nearest feature is calculated. If the distance exceeds a threshold, a unit vector in the direction of the group input is added to the feature set of that group. This process results in an approximate tiling of the group subspace, but only in the region of the subspace where observations occur. It avoids creating unnecessary features. There is currently no mechanism for forgetting features that are rarely observed, but such a mechanism may be introduced in future versions.

Creating features on subsets of the input results in features that are localized. This is in contrast to feature creation methods, such as principal components analysis (PCA), that create features spanning the entire input space.

Mapping input to features In addition to the feature learning described above, mapping the input onto features occurs at each time step as well. Input elements are broken into their respective groups and are projected onto the features of that group. The magnitude of each projection acts as a vote, and the feature with the strongest vote suppresses the others through a winner-take-all operation. The output of each group at each time step is a feature vector of $\mathbf{f} \in \mathbb{R}^q | 0 \leq f_i \leq 1$, where q is the number of features in the group, and all $f_i = 0$ except for the winning feature. The feature outputs of all groups are then combined into a master feature stimulation vector, $\mathbf{g} \in \mathbb{R}^s | 0 \leq g_i \leq 1$, where s is the total number of features across all groups.

Mapping input onto the feature set, is illustrated in Figure 2. Progressing from the bottom of the figure to the top: sensors respond the the environment, then they activate low level features, which in turn activate progressively higher level features. The set of all active features are passed to the reinforcement learner at each time step. These constitute the agent's interpretation of its environment and its current state within it. The features in the figure are displayed according to the layout of the sensors, but this is only for the convenience of the human reader. BECCA has no notion of how the sensors or features relate to each other in the world. The illustration shows visual data only, but this is also for convenience. BECCA handles information from all sensor types the same way and build features of any type, even combining multiple sensor types in a single feature if the agent's observations justify doing so.

Basic features In some tasks, the designer may wish to engineer some basic features that incorporate knowledge of the world, rather than forcing BECCA to learn them all. For

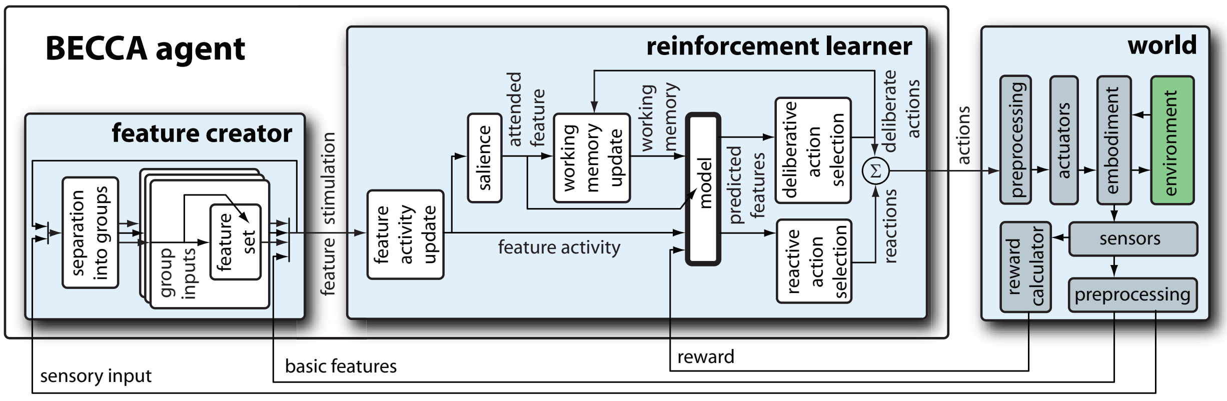

Figure 1: At each time step, the BECCA agent completes one iteration of the sensing-learning-planning-acting loop, consisting of nine major steps: 1) Reading in observations and reward. 2) Updating its feature set. 3) Expressing observations in terms of features. 4) Attending to the most salient feature. 5) Updating its model. 6) Predicting likely outcomes based on an internal model. 7) Selecting a deliberative action based on the expected reward of likely outcomes. 8) Generating reactive actions. 9) Combining deliberative and reactive actions and sending the resulting commands out to the world.

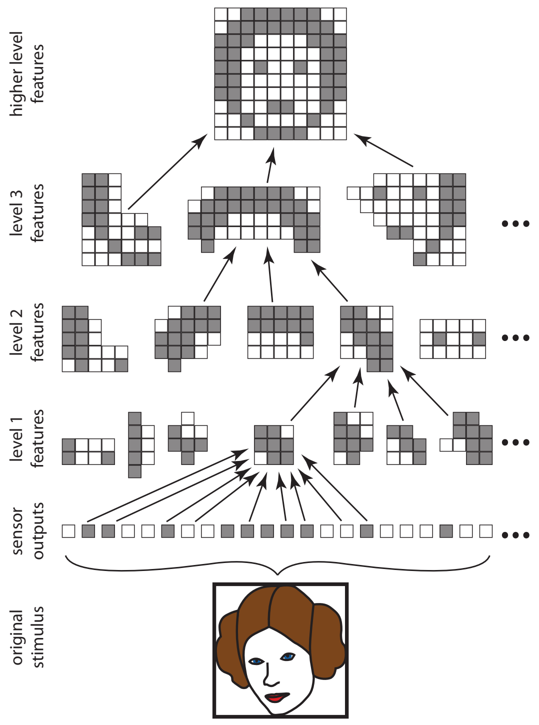

Figure 2: A conceptual illustration, using visual data, of the feature creator mapping input to previously created features. In this example, individual sensors are single pixels. BECCA had previously formed groups and identified features within those groups. The features most strongly excited by the inputs become active and excite higher level features. First level features resemble oriented line segments and points. Higher level features resemble curves, contours, and complete objects. Arrows show which features contribute most strongly to the activation of their higher level counterpart.

instance, finding blobs in vision data using a canned preprocessing step provides information to BECCA that may be useful even before it is combined with other inputs. These basic features are concatenated to the feature stimulation vector as shown in Figure 1.

One of the most significant aspects of BECCA's feature creator is that it allows for the creation of hierarchical feature sets, that is, features can be built into higher level features. Elements of the feature stimulation vector conform to the same constraints as sensory inputs and so can be fed back and treated as such. This loop lets features be treated as any other input, grouped, and built into higher level features. There is no inherent constraint on the complexity of the features that can be constructed in this way. It is driven only by structure inherent in the observations.

BECCA's feature creator is incremental, meaning that it incorporates new observations one at a time rather than in batches, and it is on-line, meaning that each observation is incorporated before the next one is received. These attributes are uncommon in feature creation algorithms and suit BECCA well to interacting with the natural world through a robotic embodiment.

Reinforcement learner

The role of the reinforcement learner is to maximize the amount of reward BECCA collects. It does this by taking in feature stimulation information, using that to form a model of the world, and using that model to select actions that are likely to result in a high payoff.

Feature processing and attention As the feature stimulation vector is passed in at each time step, it is combined with decayed versions of recent feature stimulation values by means of leaky integration to form a feature activity vector. This has the effect of giving a stimulated feature some temporal persistence across several time steps. A salience filter selects only one feature to attend. A working memory filter maintains a brief history of attended features, also using leaky integration. By this mechanism, attended features are also given a small amount of temporal persistence.

Model creation At the core of the reinforcement learner is the world model it creates. The model consists of a list of feature-space transitions in the form of *cause* - *effect* pairs, each with an associated *count* and *reward* value. At each time step, the previous working memory is compared to the list of causes and the attended feature is compared to the list of effects. If a similar pair exists within the model, its count is incremented, and its reward value is adjusted toward the current reward. If there isn't a sufficiently similar pair, the previous working memory and attended feature are added as a new cause-effect pair.

The model that results is similar to a first-order Markov model, but with important differences. At any point, the tabular entries in BECCA's model can be combined and, under a frequentist assumption, transition probabilities between causes and effects can be estimated, similar to a Markov model. However, the transition probabilities are not of transitions from state to state, but from leakily-integrated state (working memory) to state (attended feature). As a result, BECCA's model resembles a degenerate form of a higher-order Markov model more than a first-order model. It incorporates a small amount of history into each cause, allowing it to capture dynamics and complexities of structure that could not be represented in a first-order Markov model without explicitly creating additional state variables for that purpose. Also, since the basis for constructing causes and effects is an impoverished state representation (attended features only), the cause-effect transitions take on a different significance than full state transitions.

As BECCA is exposed to a variety of inputs and chooses actions over more time steps, the model accumulates more cause-effect transition pairs. The count associated with each transition establishes its frequency of observation, and the reward value represents the expected reward associated with making that transition. Each transition represents a one-step path segment through feature space. (Actually, since the attention process effectively ignores so many of the active features, a transition is a path segment through a subspace of the whole feature space. Alternatively, it can be considered as a region of a hyperplane in the full feature space.) Once the number and density of the path segments grow sufficiently high, it becomes possible for BECCA to make multi-step plans by linking several segments together to get from a current state to a desired state.

Prediction The most important capability required for planning is prediction. BECCA is designed to operate in embodiments and environments that are stochastic. The ability to predict the likely outcomes of various actions allow BECCA to choose the action with the best expected outcome. In this way, prediction, planning, and control are all part of the same process.

Prediction occurs by matching the current state (either the working memory or the feature activity in its entirety) against all the causes in the model. The strength of the matches and the count of each transition affect the weight with which each effect is predicted.

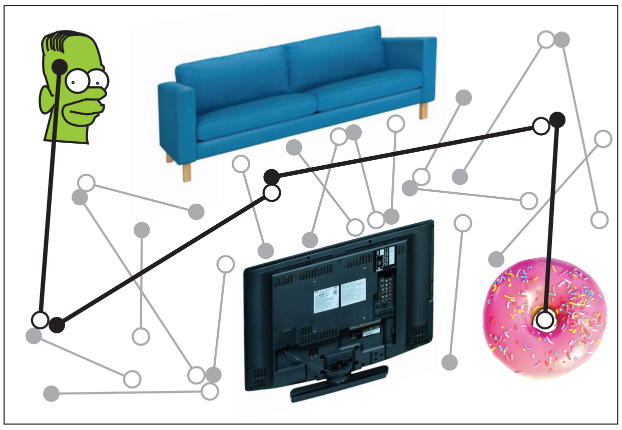

Figure 3: A conceptual illustration, using Cartesian position features, of the reinforcement learner using its model to plan a path to a rewarded state. In this example, the agent has already constructed a model of its world through exploration. Each line segment represents a cause(●)–effect(○) transition. The bold lines show the path through the feature space chosen to reach a reward.

Action selection In a typical embodied system, many effects will be conditional on the actions selected by the agent. In BECCA, action selection relies on predicted effects. Each cause-effect transition has a reward associated with it, learned by experience. The transitions with both high expected reward and high similarity to the current state are inspected to see whether they involve an agent action, and if they do, that action is executed. There are two types of action selection, reactive and deliberative. In reactive action selection, the entire feature activity vector is used to seed predictions from the model and actions are executed with a magnitude proportional to the quality of match with each transition. In deliberative action selection, the working memory only is used to seed predictions from the model and actions from a single transition, chosen both for its similarity to the current state and for its expected reward, are executed with a magnitude of one. Both types of action selection take place at each time step and the actions resulting from both are summed nonlinearly such that no actions are executed with a magnitude greater than one. The deliberative action is also fed back to the working memory so that it can be appropriately recorded as part of the cause when the model is trained on the following time step.

World

The world can be subdivided into two parts, the portion that is engineered, including the physical embodiment of the agent and the reward calculator, and the portion that is naturally occurring and presumably not under the direct control of the system designer (labeled *environment* in Figure 1). It is assumed that both parts are unknown to BECCA at the time of its creation.

Preprocessing The actions issued from BECCA and the inputs it receives are all constrained to be real valued between zero and one. In most cases, some processing is required to translate actual sensor readings to this range. For instance, pixel brightness values given as 8-bit unsigned integers may be divided by 256. Or action values may be multiplied and rounded in order to produce an integer motor step value between 0 and 1023. In other cases, more significant processing may need to be done. A continuous position sensor value might be broken into bins, with each bin represented as a separate input channel to BECCA. This would likely be used as a basic feature input to BECCA, as discussed above. Other more extreme basic feature processing might include specialized blob detection or facial recognition algorithms, speech-to-text processors, occupancy grid creators, or sensor fusion methods. Higher level preprocessing for actions could include the incorporation of coordinated multi-actuator motions, fixed motion primitives, set points for high-bandwidth low-level controllers, and even heuristic goal pursuit subroutines. Where domain knowledge can be applied to accelerate BECCA's learning, the preprocessing of the sensor and action information in the embodiment is the place to implement it. This allows the BECCA agent to remain unchanged between many different implementations, yet still allows its behavior to be customized where required.

Embodiment The physical embodiment includes the actuators and sensors of the hardware, as well as the mechanisms and physics that couple them. It is the physical manifestation of the agent. In practice, the embodiment can be simulated, with virtual sensors and actuators. In fact, once in simulation, the embodiment need not resemble a physical system at all. Virtual actuators may buy and sell stocks or send emails, and virtual sensors may report any relevant type of information. However, the focus of the research effort described here has been on robots behaving in a physical environment.

Environment The environment is everything else in the universe that is not BECCA or its embodiment. It may include elements that are abruptly non-linear, deformable, time-varying, and stochastic. It can include other robots, other BECCA agents, and other humans. In short, it can include anything that humans encounter as they interact with the world. Although few, if any, rigorous statements can be made about any agent's performance in such a broad setting, BECCA at least avoids making explicit assumptions about the environment that automatically limit its performance in the general case, such as time invariance.

Reward calculator Together with the embodiment and the environment, a reward function completes the definition of a reinforcement learning task. The reward calculator may have access to all of the sensors and the preprocessed input information. Using this information, it produces a real valued reward between -1 and 1. BECCA also avoids making assumptions about the reward function. It may be a time varying and stochastic function of any combination of sensor observations. In one type of implementation, it may be directly controlled by a human trainer. The trainer, from BECCA's standpoint, would be just one element of the environment, and the reward function would be a simple function of the sensed environmental state, however difficult to predict.

Development cycle

Given the limitations of current AI and robotics approaches, NWI is too large in scope to be useful as a near-term goal. It is helpful to select a subset of NWI tasks as a development target and expand that subset as capabilities improve. However, in this process it is critical to keep in mind the ultimate goal and avoid engineering task-specific solutions that fail to generalize to new tasks. BECCA's development cycle was designed to emphasize breadth in NWI and avoid overfitting performance to specific tasks. BECCA's success as an agent is measured by the number of tasks on which it achieves adequate performance, rather than the extent to which it achieves optimal performance on a single task.

As a first pass to narrowing down the NWI task, a challenge task was chosen: Search and Retrieve. (Rohrer 2010) In the search and retrieve task (S&R), an embodied agent is instructed to enter an environment and retrieve an object. It receives a reward when it returns with the desired object. The instruction may be of any type, including verbal, textual, visual, and/or gestural. There are no constraints on the nature of the environment, the object, or the embodiment. S&R is sufficiently general to capture much of the breadth in NWI, but specific enough to help focus laboratory research efforts.

In order to grow an S&R-relevant task set that is simple enough to be feasible yet complex enough to be interesting, a scaffolded approach to task selection has been taken. First, a trivially easy reinforcement learning problem is selected. Then the following method is applied iteratively:

1. Informed by experimental psychology and cognitive neuroscience, refine BECCA's algorithms until it performs adequately on the most recently selected task. 'Adequate' is subjective, falling somewhere between 'at chance' and 'optimal'.
2. Test BECCA on all previously selected tasks. If performance is no longer adequate on any one of them, select it and return to step 1.
3. Select a new task to add to the set. It should be relevant to the S&R task and be slightly more challenging than the previous task.
4. Return to step 1.

Results

So far a small, but nontrivial, set of tasks has been added to BECCA's repertoire using this methodology. MATLAB code for most of these tasks can be downloaded (Rohrer 2011a), and the papers cited provide technical details of the tasks and results.

- **One dimensional grid world.** As a trivial first task, BECCA was rewarded for maintaining one state in a simulated one-dimensional grid world with nine discrete states. It achieved optimal performance. In a more challenging variant, BECCA was limited to single-state steps, requiring multi-step planning.

- **Two dimensional grid world** BECCA was rewarded for spending time in one element of a 5×5 grid, and punished for spending time in another. In one variant, there was one sensor per grid element. In a second variant, there was one sensor per row and one per column.
- **One dimensional visual servoing** BECCA was allowed to saccade in one dimension across a constructed image. It was rewarded for fixating on one position. This task operated on pixellated visual input and required first creating features before the model could be populated. BECCA's performance approached optimal. (Rohrer 2011d)
- **Two dimensional visual servoing** Similar to the previous task, but in two dimensions. (Rohrer 2011c)
- **Visual feature creation** Although not a proper reinforcement learning task, this task focused on feature creation from natural images. (Rohrer 2011b) Hierarchical feature sets were created from pixellated, center-surround image patches from the Caltech-256 data set. (Griffin, Holub, and Perona 2007)
- **Auditory feature creation** This task was similar to visual feature creation, but with binned fast Fourier transforms of audio data taken from National Public Radio broadcasts. (Rohrer 2011b)
- **Mobile robot visual search** In an older form of BECCA, a mobile robot (SRV-1, Surveyor Corp.) navigated a laboratory environment, being rewarded alternately for finding low and high contrast visual scenes. (Rohrer 2009a) This behavior was described qualitatively as "hiding" and "seeking". A video documents the behavior. (Rohrer 2009b)
- **CoroBot robot arm positioning** Most recently, BECCA has been demonstrated in two additional physical robot embodiments, a Whole Arm Manipulator (WAM, Barrett Technology, Inc.) and a CoroBot (CoroWare Technologies, Inc.). Several tasks were implemented on these platforms. A robotic version of the one-dimensional grid world was implemented on the CoroBot. The arm moved between a small number of discrete positions and was rewarded for one of them. (Demonstrated at AGI 2011)
- **CoroBot hand-eye coordination** In a second CoroBot task involving vision, the robot was rewarded for occluding its field of vision with its own hand. This represented a primitive first step toward visuo-motor coordination. The CoroBot learned to perform both of these tasks optimally in the ϵ-greedy sense. (Demonstrated at AGI 2011)
- **WAM robot arm positioning with transfer from simulation** The WAM performed a 50-state, three-dimensional version of the grid world task. Due to the larger scope of the task, the bulk of the learning was done in simulation, then transferred to the hardware task. (Submitted for publication)
- **Human training of the WAM** In a second task, the WAM moved between a small number of states, while a human trainer provided reward and punishment to shape its behavior. After a brief period of experimentation, the WAM settled into the rewarded state. (Submitted for publication)

Discussion

BECCA still falls far short of its intended purpose, to become a general reinforcement learning agent capable of addressing Natural World Interaction. It has, however, made first steps in this direction and, more importantly, it lacks any built-in limitations that automatically preclude its ability to achieve that goal. BECCA's development path provides a roadmap along which progress can be measured. Future work will focus on expanding the set of tasks BECCA can address, working toward high levels of performance on the search and retrieve task.

Acknowledgments

The robot tasks were developed and supported by Pate Brogger, Sean Hendrix, and Dr. J. Dan Morrow of Sandia and by Nicholas Malone, Prof. Ron Lumia, Prof. Lydia Tapia, and Prof. John Wood of the University of New Mexico. Fredrick Rothganger and Patrick G. Xavier contributed to in depth discussions of various learning approaches during BECCA's development. This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

Beeson, P.; Modayil, J.; and Kuipers, B. 2010. Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy. *The International Journal of Robotics Research* 29(4):428–459.

Griffin, G.; Holub, A.; and Perona, P. 2007. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology. http://authors.library.caltech.edu/7694.

Konidaris, G.; Kuindersma, S.; Grupen, R.; and Barto, A. 2011. Autonomous skill acquisition on a mobile manipulator. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, 1468–1473.

Rohrer, B. 2009a. Model-free learning and control in a mobile robot. In *Proceedings of the Fifth International Conference on Natural Computation*.

Rohrer, B. 2009b. Robot learning with a biologically-inspired brain (becca). http://www.youtube.com/watch?v=mJ8LMMHsUf8.

Rohrer, B. 2010. Accelerating progress in artificial general intelligence: Choosing a benchmark for natural world interaction. *Journal of Artificial General Intelligence* 2:1:28.

Rohrer, B. 2011a. BECCA code page. http://www.sandia.gov/rohrer/code.html.

Rohrer, B. 2011b. Biologically inspired feature creation for multi-sensory perception. In *Second International Conference on Biologicall Inspired Cognitive Architectures*.

Rohrer, B. 2011c. A developmental agent for learning features, environment models, and general tasks. In *First Joint International Conference on Development and Learning and Epigenetic Robotics*.

Rohrer, B. 2011d. An implemented architecture for feature creation and general reinforcement learning. In *Workshop on Self-Programming, 4th International Conference on Artificial General Intelligence*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: MIT Press.

Sutton, R. S.; Modayil, J.; Delp, M.; Degris, T.; Pilarski, P. M.; White, A.; and Precup, D. 2011. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*.